Docket No. AUS920040057US1 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Lee**	§	
	§	Group Art Unit: **2185**
Serial No. **10/814,733**	§	
	§	Examiner: **Yaima Campos**
Filed: **March 31, 2004**	§	
	§	
For: **Data Processing System and Computer Program Product for Support of System Memory Addresses with Holes**	§ §	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on June 22, 2007.

A fee of $500.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-20

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 2-3, 8-9, 19.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1, 4-7, 10-18, 20.
4. Claims allowed: None.
5. Claims rejected: 1-20.
6. Claims objected to: None.

C. CLAIMS ON APPEAL

The claims on appeal are: 1, 4-7, 10-18, 20.

STATUS OF AMENDMENTS

First Office Action was mailed May 2, 2006; Response to First Office Action was filed May August 2, 2006; Second Office Action was mailed October 19, 2006; Response to Second Office Action was filed January 19, 2007; Final Office Action was mailed March 23, 2007; Response to Final Office Action was filed May 23, 2007; an Advisory Action was mailed June 6, 2007; a Notice of Appeal was filed June 22, 2007. A Supplemental Response to Final Office Action is filed herewith to correct a typographical error.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method of supporting memory addresses with holes. A first physical address range that is allocated for system memory for an operating system run by a processor configured to support logical partitioning is virtualized to produce a first logical address range. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #210; Figure 4 #410; and page 11, Table A.)

A second physical address range that is allocated for system memory for the operating system is virtualized to produce a second logical address range, wherein the first physical address range and the second physical address range are non-contiguous and the first logical address range and the second logical address range are contiguous. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #211; Figure 4 #410; and page 11, Table A.)

A memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range is virtualized to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges. (Specification page 8, lines 16-24; page 10, lines 11-18; Figure 2 #220; Figure 4 #420; and page 11, Table A.)

The steps of virtualizing the first physical address range, the second physical address range, and the memory mapped input/output physical address range comprise maintaining a mapping table that defines physical addresses and corresponding logical addresses. (Specification page 9, lines 16-25.)

Maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges. (Specification page 9, lines 16-25.)

B. CLAIM 6 - DEPENDENT

Claim 6 depends from claim 1 and describes the memory mapped input/output physical address range being allocated for cache inhibited addresses. (Specification page 8, lines 15-17.)

C. CLAIM 7 - INDEPENDENT

The subject matter of claim 7 is directed to a computer program product that is stored in a computer readable medium for virtualizing non-contiguous physical memory ranges into a contiguous logical address range. (Specification page 13, line 14, through page 14, line 2.)

Instructions are included for virtualizing a first range of contiguous physical addresses, which are allocated for system memory for an operating system run by a processor configured to support logical partitioning, to produce a first range of contiguous logical addresses. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #210; Figure 4 #410; and page 11, Table A.)

Instructions are included for virtualizing a second range of contiguous physical addresses, which are allocated for system memory for the operating system, to produce a second range of contiguous logical addresses, the first range of contiguous physical addresses and the second range of contiguous physical addresses being non-contiguous, the first range of contiguous logical addresses and the second range of contiguous logical addresses being contiguous and forming a combined range of contiguous logical addresses. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #211; Figure 4 #410; and page 11, Table A.)

Instructions are included for virtualizing a third range of contiguous physical addresses, which is allocated for memory mapped input/output, that is intermediate to the first range of contiguous physical addresses and the second range of contiguous physical addresses to produce a third range of contiguous logical addresses, a lowermost logical address of the third range of contiguous logical addresses exceeding an uppermost logical address of the combined range of contiguous logical addresses. (Specification page 8, lines 16-24; page 10, lines 11-18; Figure 2 #220; Figure 4 #420; and page 11, Table A.)

Instructions are included for maintaining a mapping table that defines physical addresses and their corresponding logical addresses. (Specification page 9, lines 16-25.)

The mapping table is maintained in at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses. (Specification page 9, lines 16-25.)

D. CLAIM 14 - DEPENDENT

Claim 14 depends from claim 7 and describes the third range of contiguous physical addresses being allocated for cache inhibited memory mapped input/output addresses. (Specification page 8, lines 15-17.)

E. CLAIM 15 - INDEPENDENT

The subject matter of claim 15 is directed to a data processing system for supporting non-contiguous system memory ranges.

A memory is included that contains a first range of contiguous physical addresses allocated for system memory, a second range of contiguous physical addresses allocated for system memory, and a third range of contiguous physical addresses allocated for memory-mapped input/output, the third range of contiguous physical addresses intermediate to the first range of contiguous physical addresses and the second range of contiguous physical memory addresses. (Specification page 7, line 24, through page 8, line 23; Figure 1 #109; and Figure 2 #200.)

The first range of contiguous physical addresses and the second range of contiguous physical addresses are non-contiguous; (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #210-211; Figure 4 #410; and page 11, Table A.)

A processor is included for virtualizing the first range of contiguous physical addresses to produce a first range of contiguous logical addresses. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 1 #102/104; Figure 2 #210; Figure 4 #410; and page 11, Table A.)

The processor is for virtualizing the second range of contiguous physical addresses to produce a second range of contiguous logical addresses. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 2 #211; Figure 4 #410; and page 11, Table A.)

The first range of contiguous logical addresses and the second range of contiguous logical addresses are contiguous and forming a combined range of contiguous logical addresses. (Specification page 8, lines 1-7; page 10, line 22, through page 11, line 2; Figure 4 #410; and page 11, Table A.)

The processor is for virtualizing the third range of contiguous physical addresses to produce a third range of contiguous logical addresses, a lowermost logical address of the third range of

contiguous logical addresses exceeding an uppermost logical address of the combined range of contiguous logical addresses. (Specification page 8, lines 16-24; page 10, lines 11-18; Figure 4 #420; and page 11, Table A.)

A set of instructions are executed by the processor for virtualizing the first, second, and third ranges of contiguous physical addresses, wherein the set of instructions is maintained in the memory in at least one of the first and second ranges of contiguous physical addresses. (Specification page 9, lines 16-25.)

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

1. Whether claims 1-5, 7-13, and 15-20 are obvious over *Arndt*, Logical Partitioning Via Hypervisor Mediated Address Translation, U.S. Patent No. 6,877,158, dated April 5, 2005 (hereinafter referred to as "*Arndt*") and *Stine* et al., Memory Allocation System, U.S. Patent No. 6,629,111, dated September 30, 2003 (hereinafter referred to as "*Stine*") under 35 U.S.C. § 103(a); and

2. Whether claims 6 and 14 are obvious over *Arndt* in view of *Stine* and further in view of *Yazdy* et al., Cache Management During Cache Inhibited Transaction for Increasing Cache Efficiency, U.S. Patent No. 6,256,710, dated July 3, 2001 (hereinafter referred to as "*Yazdy*") under 35 U.S.C. § 103(a) .

ARGUMENT

A. SUPPLEMENTAL RESPONSE TO FINAL OFFICE ACTION

A Supplemental Response to Final Office Action is filed herewith to correct a typographical error in claim 1. The claims appended to this Appeal Brief reflect the amended claim.

B. GROUND OF REJECTION 1 (Claims 1-5, 7-13, and 15-20)

The Examiner rejected claims 1-5, 7-13, and 15-20 under 35 U.S.C. §103(a) as being obvious over U.S. Patent 6,877,158 issued to *Arndt* in view of U.S. Patent 6,629,111 issued to *Stine* et al. This position is not well-founded.

The combination of *Arndt* and *Stine* does not render Appellant's claims 1-5 obvious because the combination does not teach or suggest the combination of virtualizing a first physical address range allocated for system memory for an operating system run by a processor configured to support logical partitioning to produce a first logical address range; and virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges.

The combination of *Arndt* and *Stine* also does not render Appellant's claims 1-5 obvious because the combination does not teach or suggest wherein the steps of virtualizing the first physical address range, the second physical address range, and the memory mapped input/output physical address range comprise maintaining a mapping table that defines physical addresses and corresponding logical addresses; and wherein maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges.

The Examiner stated that *Arndt* does not teach "virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges" and relies on *Stine* to supply this feature.

Appellant respectfully disagrees that *Stine* teaches this feature.

Stine teaches using a translation look-aside buffer (TLB) which converts virtual addresses to physical addresses. A library can be loaded into one or more pages of physical memory. Once loaded, each page is accessed by a virtual address using the TLB.

Stine teaches a method of allocating memory in systems that restrict the mapping between physical addresses and virtual addresses to those existing on page boundaries. Virtual memory is allocated to a library when the library is to be accessed. This virtual memory is allocated in pages. Since a library typically does not occupy exactly a page, there is a portion of the page (which had already been allocated) that is unused. Thus, a "hole" exists in a virtual memory page when a library, which is loaded into a virtual page, is not large enough to occupy the entire virtual page. Similarly, there is a corresponding "hole" in the physical memory page that was mapped to the virtual page.

Stine takes advantage of the empty virtual memory storage that is left when a library does not occupy the entire virtual page that was allocated for that library. When another library needs to be loaded into virtual memory, *Stine* searches for an available "hole" in the allocated virtual memory. If a hole is found that is the same size or larger than the library that needs to be loaded, the library is loaded into the existing hole. Thus, in *Stine*, two different libraries can be loaded into the same page. See *Stine*, column 10, lines 30-51.

The new library is addressed using a virtual address that is calculated by adding together the virtual address of the original library already loaded into the page plus the size of the original library. Thus, the virtual address that is used to address the new library had already been allocated, i.e. produced, to the existing library that was already loaded into the page. In this manner, no new virtual addresses need to be produced when the new library is loaded into memory. *Stine* makes this point clear utilizing Figure 10 and its related description in column 10, lines 9-29.

Stine teaches allocating a memory segment from physical memory for a library. The TLB maps virtual address to the physical addresses of this memory segment. *Stine* then teaches this memory segment including a "hole", which is an unused portion of the memory segment. A second library could be stored in this hole, instead of allocating another memory segment for this second library. In this case where the second library is stored in the hole, additional physical memory/virtual memory is not allocated for the second library. Because additional physical

memory/virtual memory is not allocated for the hole and a third logical address range is not produced as a result of virtualizing memory, *Stine* does not teach virtualizing a physical address range to produce a third logical address range.

Stine does not teach "virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges".

Stine does not teach virtualizing physical memory of one of these "holes". *Stine* searches for a hole in the allocated virtual address space. If a hole is found, that hole is in a virtual memory page that was already mapped to physical memory. Therefore, nothing is virtualized to produce the hole. A third logical address range is not produced as a result of virtualizing a physical address range of the hole. The virtualization of physical memory to virtual memory had already been completed. *Stine* teaches merely a reassignment of virtual addresses, which had already been produced, from an existing library to a new library.

> If it is determined at block 801 that the library fits in an existing memory segment (e.g., hole in an existing memory segment), virtual memory need not be allocated for the library. In other words, <u>virtual memory need not be allocated for a hole</u>.

[Emphasis Added]

Stine, column 9, lines 40-44.

Stine, Figure 10, provides a good depiction of the differences between *Stine* and Appellant's claims. Figure 10 depicts a memory segment 1000. This entire memory segment 1000 was allocated/virtualized at one time for the library to produce one logical address range, which is the entire virtual address range depicted in Figure 10. That is, the memory segment 1000 was not allocated in pieces. For example, memory from 0x1000 to 0x11000 was allocated at the same time memory from 0x11000 to 0x11800 was allocated, and at the same time memory from 0x11800 to 0x11C00 was allocated. Thus, portion 1002 of memory segment 1000 was allocated at the same time portion 1004 of memory segment 1000 was allocated. That is, they were not virtualized in separate steps to produce different logical address ranges.

In contradistinction, Appellant claims virtualizing a first physical address range to produce a first logical address range. Appellant also claims virtualizing a memory mapped

input/output address range to produce a third logical address range. This is a separate step from virtualizing a first physical address range to produce a first logical address range.

Stine does not teach a third logical address range as it is claimed by Appellant. In *Stine*, the logical address range of the hole, into which the new library is loaded, is not a third logical address range. The starting virtual address for the new library is the starting virtual address of the hole. This is made clear in *Stine* in Figure 10. *Stine*, column 10, lines 23-26, states: "More particularly, the starting address of the library is obtained by adding the starting virtual address of the previously loaded library to the size of the previously loaded library." Thus, the logical address of the hole was not produced as a result of virtualizing a physical address range. Therefore, the logical address of the hole cannot be a third logical address range.

Stine does not teach "wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges". As discussed above, existing logical addresses are used for the new library that is loaded into the hole. Therefore, there is no lowermost logical address of the third logical address range because there is no third logical address range.

Stine does teach allocating a new virtual address range when an existing hole does not exist. When a new library needs to be loaded and there is not an existing hole that is large enough into which the new library can be loaded, a new virtual page must be allocated. This new virtual page is allocated starting at the next available address in the virtual memory. See *Stine*, column 10, lines 2-5.

While *Stine* does teach allocating the virtual memory at the next available address, *Stine* does not teach the next available address having a lowermost logical address that exceeds a respective uppermost logical address of the first and second logical address ranges. The next available virtual address could have an address that is lower than the other virtual addresses. Nothing in *Stine* teaches the next available address exceeding the virtual addresses in which the other libraries are loaded.

For the reasons given above, the combination of *Arndt* and *Stine* does not render Appellant's claims obvious because the combination does not teach or suggest the combination of virtualizing a first physical address range allocated for system memory for an operating system run by a processor configured to support logical partitioning to produce a first logical address range; and virtualizing a memory mapped input/output physical address range that is intermediate the first

physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges.

The combination of *Arndt* and *Stine* also does not render Appellant's claims obvious because the combination does not teach or suggest wherein the steps of virtualizing the first physical address range, the second physical address range, and the memory mapped input/output physical address range comprise maintaining a mapping table that defines physical addresses and corresponding logical addresses; and wherein maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges.

According to Appellant's claims, the mapping table is maintained in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges. Thus, the mapping table is maintained in a physical address space, which is unavailable to the operating system, that is allocated to a physical address range that is virtualized for that operating system.

According to Appellant's claims, the physical address space in which the mapping table is maintained is allocated to one of the first and second physical address ranges. According to the Advisory Action that was mailed June 6, 2007, Appellant understands the Examiner to believe the page frame tables 320-350 are analogous to the mapping table claimed by Appellant. Appellant also understands the Examiner to be equating a physical resource 360 with the physical address ranges claimed by Appellant. Assuming, for the sake of argument, that a page frame table is indeed analogous to a mapping table and physical resources are analogous to physical address ranges, then there must be a physical address space allocated to one of the physical resources 360 in which a page frame table is maintained. Further, the physical address space must be unavailable to the operating system that is accessing the physical resource 360. *Arndt* does not teach this.

Arndt teaches page frame tables 320-350 containing mappings for virtual pages for an operating system image to a physical resource 361-371. Virtual memory is a method of simulating more memory for an operating system than actually exists allowing platform 200 to

run larger software programs or more programs concurrently. Virtual memory breaks up the software program into small segments, called pages, and brings as many pages into memory 240-246 that fit into a reserved area for that software program. See *Arndt*, column 4, lines 40-57. Thus, there is a part of an operating system's memory 240-246 that is reserved for a program, which means not all of the memory 240-246 is necessarily reserved and virtualized for the program. *Arndt* suggests that other physical resources could be available that could be virtualized for other programs or used for other purposes, such as for storing a page frame table. Therefore, *Arndt* does not teach a page frame table being stored or maintained in one of the physical resources. A page frame table could be stored in an area of memory 240-246 that is not reserved for that software program.

In the Advisory Action that was mailed June 6, 2007, the Examiner stated that "Appellant should note that since each OS image/logical partition has a separate page frame table (for virtual to physical address translation, which occupies a portion of physical memory) and physical resources assigned to it, the combination of the physical memory assigned to this page frame table and the physical memory assigned to physical resources comprises the total physical address space allocated to each of the different address ranges/logical partitions/OS images; therefore, maintaining a mapping table in a physical address space allocated to each physical address range and disclosing 'maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges' as claimed." Appellant respectfully disagrees.

Arndt teaches reserving an area of an operating system's memory 240-246 for a software program, which is broken into pages, and thus virtualized. *Arndt* does not teach or suggest reserving all of its memory 240-246 to be virtualized leaving none to be used for any other purpose.

In addition, nothing in *Arndt* teaches or suggests the page frame table being stored in one of the physical resources 360. The page frame table could be stored in a physical resource that is not virtualized. The Examiner assumes that all physical resources allocated to the operating system are virtualized, and thus, have a page frame table that contains translations of virtual addresses to physical addresses. This is not necessarily the case.

Also according to Appellant's claims, the physical address space allocated to a physical address range is unavailable to the operating system accessing the physical address range. To

teach this feature, *Arndt* would need to teach storing a page frame table in a space of one of the physical resources, and then making this space unavailable to the operating that is accessing that physical resource. *Arndt* does not teach this.

Arndt teaches an OS needing to access a specific physical resource. The OS makes a request to the hypervisor to access the resource. The hypervisor checks to make sure the specific physical resource is allocated for the OS. If the specific physical resource has been allocated to the OS, the hypervisor completes the virtual to real mapping for the resource, and the OS can access the specific physical resource. Therefore, the OS can access <u>all</u> physical resources for which the mapping has been completed. This is not what Appellant claims.

Appellant claims first virtualizing the physical address range for an operating system. Thus, the "mapping" takes place first. Then, Appellant claims maintaining the mapping table in a physical address space allocated to the physical address range, wherein the physical address space is unavailable to the operating system. Therefore, the operating system does not have the physical address space available to it even though the physical address range was virtualized for it. This situation could not occur in *Arndt*. In *Arndt*, if mapping has been completed for a physical resource, the OS can access it.

The combination of *Arndt* and *Stine* does not render Appellant's claim 1-5 obvious because the combination does not teach or suggest:

virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges;

wherein the steps of virtualizing the first physical address range, the second physical address range, and the memory mapped input/output physical address range comprise maintaining a mapping table that defines physical addresses and corresponding logical addresses; and

wherein maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges.

Appellant's claim 7 recites: wherein the mapping table is maintained in at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses.

As discussed above, nothing in *Arndt* teaches or suggests maintaining a page table in one of the physical resources 360. Since *Arndt* suggests the possibility that there are other resources, such as other areas of memory not included in physical resources 360 in which the page table could be stored, *Arndt* does not teach or suggest wherein the mapping table is maintained in at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses. *Stine* does not cure this deficiency of *Arndt*. Thus, the combination of *Arndt* and *Stine* does not render Appellant's claim 7 obvious.

Appellant's claim 15 recites: "a set of instructions that is executed by the processor for virtualizing the first, second, and third ranges of contiguous physical addresses, wherein the set of instructions is maintained in the memory in at least one of the first and second ranges of contiguous physical addresses".

As discussed above, nothing in *Arndt* teaches or suggests maintaining a page table in one of the physical resources 360. Since *Arndt* suggests the possibility that there are other resources, such as other areas of memory not included in physical resources 360 in which the page table could be stored, *Arndt* does not teach or suggest a set of instructions that is executed by the processor for virtualizing the first, second, and third ranges of contiguous physical addresses, wherein the set of instructions is maintained in the memory in at least one of the first and second ranges of contiguous physical addresses. *Stine* does not cure this deficiency of *Arndt*. Thus, the combination of *Arndt* and *Stine* does not render Appellant's claim 15 obvious.

The remaining claims depend from one of the independent claims discussed above and are patentable for the reasons given above. Therefore, the rejection of claims 1-5, 7-13 and 15-20 under 35 U.S.C. § 103(a) has been overcome.

C. GROUND OF REJECTION 2 (Claims 6 and 14)

The Examiner rejected claims 6 and 14 under 35 U.S.C. §103(a) as being obvious over U.S. Patent 6,877,158 issued to *Arndt* in view of U.S. Patent 6,629,111 issued to *Stine* et al, and further in view of U.S. Patent 6,256,710 issued to *Yazdy*. This position is not well-founded.

Claim 6 depends from claim 1. Appellant's claim 6 recites: "wherein the memory mapped input/output physical address range is allocated for cache inhibited addresses".

Claim 14 depends from claim 7. Appellant's claim 14 recites: "wherein the third range of

contiguous physical addresses is allocated for cache inhibited memory mapped input/output addresses".

The Examiner states that the combination of *Arndt* and *Stine* does not teach the physical addresses being allocated for cache inhibited memory mapped input/output addresses. The Examiner relies on *Yazdy* to teach this feature.

Yazdy teaches an area of main memory that is non-cacheable. See *Yazdy*, column 2, lines 23-14. *Yazdy* does not, however, teach virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges, wherein the memory mapped input/output physical address range is allocated for cache inhibited addresses. Therefore, the combination of *Arndt*, *Stine*, and *Yazdy* does not render Appellant's claim 6 obvious.

Yazdy does not teach instructions for virtualizing a third range of contiguous physical addresses, which is allocated for memory mapped input/output, that is intermediate to the first range of contiguous physical addresses and the second range of contiguous physical addresses to produce a third range of contiguous logical addresses, a lowermost logical address of the third range of contiguous logical addresses exceeding an uppermost logical address of the combined range of contiguous logical addresses, wherein the third range of contiguous physical addresses is allocated for cache inhibited memory mapped input/output addresses. Therefore, the combination of *Arndt*, *Stine*, and *Yazdy* does not render Appellant's claim 14 obvious.

D. CONCLUSION

Appellant believes the claims are allowable over the cited prior art and that the application is in condition for allowance. Accordingly, Appellant respectfully requests the Board of Patent Appeals and Interferences to overturn the rejections set forth in the Final Office Action.

/Lisa L.B. Yociss/
Lisa L.B.Yociss
Reg. No. 36,975
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

<u>**CLAIMS APPENDIX**</u>

The text of the claims involved in the appeal are:

1. A method of supporting memory addresses with holes, the method comprising the computer implemented steps of:

virtualizing a first physical address range allocated for system memory for an operating system run by a processor configured to support logical partitioning to produce a first logical address range;

virtualizing a second physical address range allocated for system memory for the operating system to produce a second logical address range, wherein the first physical address range and the second physical address range are non-contiguous and the first logical address range and the second logical address range are contiguous;

virtualizing a memory mapped input/output physical address range that is intermediate the first physical address range and the second physical address range to produce a third logical address range, wherein a lowermost logical address of the third logical address range exceeds a respective uppermost logical address of the first and second logical address ranges;

wherein the steps of virtualizing the first physical address range, the second physical address range, and the memory mapped input/output physical address range comprise maintaining a mapping table that defines physical addresses and corresponding logical addresses; and

wherein maintaining the mapping table further comprises maintaining the mapping table in a physical address space allocated to one of the first and second physical address ranges, and wherein the physical address space is unavailable to the operating system accessing the first and second physical address ranges.

4. The method of claim 1, wherein the third logical address range is non-contiguous with the first logical address range and the second logical address range.

5. The method of claim 1, further comprising:

allocating a portion of at least one of the first physical address range and the second physical address range for a logical partitioning management software layer.

6. The method of claim 1, wherein the memory mapped input/output physical address range is allocated for cache inhibited addresses.

7. A computer program product that is stored in a computer readable medium for virtualizing non-contiguous physical memory ranges into a contiguous logical address range, the computer program product comprising:

instructions for virtualizing a first range of contiguous physical addresses, which are allocated for system memory for an operating system run by a processor configured to support logical partitioning, to produce a first range of contiguous logical addresses;

instructions for virtualizing a second range of contiguous physical addresses, which are allocated for system memory for the operating system, to produce a second range of contiguous logical addresses, the first range of contiguous physical addresses and the second range of contiguous physical addresses being non-contiguous, the first range of contiguous logical addresses and the second range of contiguous logical addresses being contiguous and forming a combined range of contiguous logical addresses;

instructions for virtualizing a third range of contiguous physical addresses, which is

allocated for memory mapped input/output, that is intermediate to the first range of contiguous physical addresses and the second range of contiguous physical addresses to produce a third range of contiguous logical addresses, a lowermost logical address of the third range of contiguous logical addresses exceeding an uppermost logical address of the combined range of contiguous logical addresses;

instructions for maintaining a mapping table that defines physical addresses and their corresponding logical addresses; and

wherein the mapping table is maintained in at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses.

10. The computer program product of claim 7, further comprising instructions for converting a logical physical address into a corresponding physical address.

11. The computer program product of claim 7, further comprising:

instructions for converting a logical physical address into a corresponding physical address; and

the instructions for converting a logical physical address into a corresponding physical address being maintained in at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses.

12. The computer program product of claim 7, wherein the third range of contiguous logical addresses and the combined range of contiguous logical addresses are non-contiguous.

13. The computer program product of claim 12, further comprising:

instructions for allocating a portion of at least one of the first range of contiguous physical addresses and the second range of contiguous physical addresses for a logical partitioning management software layer.

14. The computer program product of claim 7, wherein the third range of contiguous physical addresses is allocated for cache inhibited memory mapped input/output addresses.

15. A data processing system for supporting non-contiguous system memory ranges, comprising:

a memory that contains a first range of contiguous physical addresses allocated for system memory, a second range of contiguous physical addresses allocated for system memory, and a third range of contiguous physical addresses allocated for memory-mapped input/output, the third range of contiguous physical addresses intermediate to the first range of contiguous physical addresses and the second range of contiguous physical memory addresses;

the first range of contiguous physical addresses and the second range of contiguous physical addresses being non-contiguous;

a processor for virtualizing the first range of contiguous physical addresses to produce a first range of contiguous logical addresses;

the processor for virtualizing the second range of contiguous physical addresses to produce a second range of contiguous logical addresses;

the first range of contiguous logical addresses and the second range of contiguous logical addresses being contiguous and forming a combined range of contiguous logical addresses;

the processor for virtualizing the third range of contiguous physical addresses to produce a third range of contiguous logical addresses, a lowermost logical address of the third range of contiguous logical addresses exceeding an uppermost logical address of the combined range of contiguous logical addresses;

a set of instructions that is executed by the processor for virtualizing the first, second, and third ranges of contiguous physical addresses, wherein the set of instructions is maintained in the memory in at least one of the first and second ranges of contiguous physical addresses.

16. The data processing system of claim 15, further comprising a data set, wherein the data set is a mapping table defining logical-to-physical memory address translations.

17. The data processing system of claim 15, further comprising a set of instructions that is executed by the processor, wherein the set of instructions provides logical partitioning management.

18. The data processing system of claim 15, further comprising a mapping table that defines logical-to-physical memory address translations, the mapping table maintained in the memory in at least one of the first and second ranges of contiguous physical addresses.

20. The data processing system of claim 15, further comprising:

the combined range of contiguous logical addresses being non-contiguous with the third range of contiguous logical addresses.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.